Beehive Web Service Tutorial

Table of contents

1. Introduction

This tutorial introduces you to the basic development cycle for Beehive web services. The tutorial assumes that you are working on a Windows machine. But, with a little common sense, it is easy to execute the tutorial on a Unix machine. For example, when you are asked to run the file `beehiveUser.cmd`, run the file `beehiveUser.sh` instead.

Tutorial Goals

In this tutorial, you will learn:

- how to create a basic Beehive web service application.
- how to use (JSR 175 and 181) metadata annotations.
- how to deploy and test a web service on Tomcat.

2. Step 1: Begin the Web Service Tutorial

2.1. To Set up the Development Environment

Complete all of the necessary and optional steps in the following topic: Beehive Installation and Setup (../setup.html)

After completing the instructions, leave the command shell open to use throughout this tutorial.

Before proceding, confirm that you have the following variables set in your shell:

- `ANT_HOME`
- `JAVA_HOME`
- `CATALINA_HOME`

Also ensure that the following elements are on your `PATH`:

- `ANT_HOME/bin`
- `JAVA_HOME/bin`

2.2. To Make a Project Folder Using the Web Service Application Template

On your C: drive, create a directory called `beehive_projects`.

Copy the folder `<BeehiveRoot>/samples/`**`wsm-blank`** into `C:/`**`beehive_projects`**.

> **Note:**
> <BeehiveRoot> refers to the top-level directory of your Beehive installation. A typical value for <BeehiveRoot> is `C:/apache/apache-beehive-1.0`.

Rename the folder `C:/beehive_projects/wsm-blank` as `C:/beehive_projects/ws_tutorial`

Before proceeding, confirm that the following directory structure exists:

```
C:
  beehive_projects
    ws_tutorial
      WEB-INF
      happyaxis.jsp
      index.html
```

2.3. Edit the build.properties File

In this section you will edit the `build.properties` file--the file that sets the build-related properties for your web service application.

Open the file `C:/beehive_projects/ws_tutorial/WEB-INF/src/build.properties` in a text editor.

Edit the `beehive.home` property so it points to the top-level folder of your beehive installation.

Edit the `service.name` property so it has the value `tutorial`. (as shown below).

For example, if your beehive installation resides at `C:/apache/apache-beehive-1.0`, then your `build.properties` file would appear as follows.

```
beehive.home=C:/apache/apache-beehive-1.0

service.name=tutorial
```

Note:

Windows users must use forwardslashes (/) not backslashes (\) in the `build.properties` file.

2.4. To Start the Tomcat Server

At the command prompt, enter:

```
%CATALINA_HOME%\bin\startup.bat
```

3. Step 2: Run the Web Service Template

3.1. To Examine the Blank.java Web Service

You are now ready to compile and run the template web service already included in the project folder--but before we run the web service, let's first look at the code.

In a text editor of your choice, open the file `C:/beehive_projects/ws_tutorial/WEB-INF/src-ws/web/Blank.java`.

The code looks like this:

```
package web;
...
import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public class Blank {

    @WebMethod
    public String sayHelloWorld(String s) {
        return "Hello world, " + s + "!";
    }
}
```

If you are familiar with Java code, everything probably looks familiar to you, although these two elements may be new:

```
    @WebService
    @WebMethod
```

`@WebService` and `@WebMethod` are "metadata annotations", a.k.a. "annotations". Annotations allow you to set properties on Java classes and methods. They can be used to generate compile-time artifacts such as configuration files or Java classes (this is how many Beehive Control annotations work) or to determine some runtime behavior (this is how Beehive Web Service annotations work).

`@WebService` annotates (or "decorates") the class Blank: this tells the runtime that Blank is a web service that listens for SOAP messages and responds in kind.

`@WebMethod` annotates the method sayHelloWorld(): this tells the runtime that the method can be invoked over the web.

3.2. To Compile the Web Service and Deploy it to Tomcat

You are now ready to compile the web service and deploy it to Tomcat.

At the command prompt, enter:

```
ant
  -f C:\beehive_projects\ws_tutorial\WEB-INF\src\build.xml
  clean
  build
```

```
  war

Copy and Paste version:
ant -f C:\beehive_projects\ws_tutorial\WEB-INF\src\build.xml clean build
war
```

Note:

The 'deploy' target builds a WAR file named tutorialWS.war and saves it at C:/beehive_projects.

To copy the WAR file to the Tomcat server, enter the following command:

```
copy C:\beehive_projects\tutorialWS.war %CATALINA_HOME%\webapps /Y
```

3.3. To Run the Web Service

Visit the index.jsp page: http://localhost:8080/tutorialWS/index.html.

Click the "Validate" link for an evaluation of the resources available to your web service. Note that you will need to download additional resources to take full advantage of Beehive web services. For example, for Axis to work properly with SOAP attachments, additional, external jars (activation.jar and mailapi.jar) are required. You will download those resources in later steps in the tutorial.

Click the "WSDL" link to see the web service's WSDL.

Click the "sayHelloWorld()" link to see a SOAP response from the web service's sayHelloWorld() method.

4. Step 3: Add a Parameterized Method to the Web Service

In this step you will add a new method to the web service: a method that accepts in 'IN' parameter.

4.1. To Edit the JWS File

Edit the file `C:/beehive_projects/ws_tutorial/WEB-INF/src-ws/web/Blank.java` so it appears as follows. Code to add appears in bold type.

```
package web;
...
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.WebParam;

@WebService
```

```
public class Blank {

    @WebMethod
    public String sayHelloWorld(String s) {
        return "Hello world, " + s + "!";
    }

    @WebMethod
    public String sayHelloWorldInParam( @WebParam String greetee )
    {
        if( greetee.equals("") )
            { greetee = "World"; }

        return "Hello, " + greetee + "!";
    }
}
```

The `@WebParam` you just added lets you pass a String parameter to the method over the web.

4.2. To Recompile and Redeploy the Web Service

Recompile and redeploy the web service using the same Ant command from step 2.

4.3. To Test the Parameterized Method

Enter the following URL in the address bar of your browser.

http://localhost:8080/tutorialWS/web/Blank.jws?method=sayHelloWorldInParam&greetee=World

The following SOAP response appears in the browser:

```
<soapenv:Envelope>
  <soapenv:Body>
    <sayHelloWorldInParamResponse>
      <ns1:result>Hello, World!</ns1:result>
    </sayHelloWorldInParamResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

5. Step 4: Add a Non-Web Invokable Method

5.1. To Edit the JWS File

Edit the file `C:/beehive_projects/ws_tutorial/WEB-INF/src-ws/web/Blank.java` so it appears as follows. Code to add appears in bold type.

```
package web;
...
```

```
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.WebParam;

@WebService
public class Blank {

    @WebMethod
    public String sayHelloWorld(String s) {
        return "Hello world, " + s + "!";
    }

    @WebMethod
    public String sayHelloWorldInParam( @WebParam String greetee )
    {
        if( greetee.equals("") )
            { greetee = "World"; }

        return "Hello, " + greetee + "!";
    }

    public String sayNothingOverTheWeb()
    {
        return "Not for Web consumption!";
    }
}
```

Note that the method added, sayNothingOverTheWeb(), does not have the annotation @WebMethod, indicating that it cannot be invoked by SOAP messages over the web.

5.2. To Recompile and Redeploy the Web Service

Recompile and redeploy the web service using the same Ant command from step 2.

5.3. To Test the Non-Web Invokable Method

Enter the following URL in the address bar of your browser.

http://localhost:8080/tutorialWS/web/Blank.jws?method=sayNothingOverTheWeb

The following SOAP response appears in the browser, indicating that the method sayNothingOverTheWeb() cannot be invoked through the web service (although it can be called by other methods within the web service).

```
<soapenv:Envelope>
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>ns1:Client</faultcode>
      <faultstring>No such operation 'sayNothingOverTheWeb'</faultstring>
      <detail>
        <ns2:hostname>[your machine name]</ns2:hostname>
```

```
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

6. Step 5: Change the SOAP Style

6.1. To Edit the JWS File

The default SOAP style for JSR-181 web services is DOC-literal. In this step you will change the style to RPC-encoded.

Edit the file
`C:/beehive_projects/ws_tutorial/WEB-INF/src-ws/web/Blank.java`
so it appears as follows. Code to add appears in bold type.

```
package web;
...
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

@WebService
@SOAPBinding(style = SOAPBinding.Style.RPC, use = SOAPBinding.Use.ENCODED)
public class Blank
{
    @WebMethod
    public String sayHelloWorld(String s){
        return "Hello world, " + s + "!";
    }

    @WebMethod
    public String sayHelloWorldInParam( @WebParam String greetee )
    {
        if( greetee.equals("") )
            { greetee = "World"; }

        return "Hello, " + greetee + "!";
    }

    public String sayNothingOverTheWeb()
    {
        return "Not for Web consumption!";
    }
}
```

6.2. To Recompile and Redeploy the Web Service

Recompile and redeploy the web service using the same Ant command from step 2.

6.3. To Test the SOAP-encoded Style

Enter the following URL in the address bar of your browser.

http://localhost:8080/tutorialWS/web/Blank.jws?method=sayHelloWorldInParam&greetee=World

The following SOAP response appears in the browser. Compare the RPC style below with the DOC style above.

```
<soapenv:Envelope>
  <soapenv:Body>
    <sayHelloWorldInParamResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <ns1:result xsi:type="xsd:string">Hello, World!</ns1:result>
    </sayHelloWorldInParamResponse>
  </soapenv:Body>
</soapenv:Envelope>
```